DENSO Robotics

THIRD PARTY PRODUCTS

【サードパーティ プロダクト】

メーカ

パナソニック デバイスＳＵＮＸ(株) 製

製品／シリーズ

ビジョンセンサ

MODEL:PV シリーズ

はじめに

本書は、「パナソニック デバイスＳＵＮＸ(株)製・ビジョンセンサ・ＰＶシリーズ」をデンソーロボットコントローラRC8シリーズと接続して使用するためのプロバイダの取扱説明書です。古いＰＶに関しては一部使用できない機能があります。接続機器の詳細および取扱いは、「パナソニック デバイスＳＵＮＸ(株)製・ビジョンセンサ・ＰＶシリーズ」の取扱説明書をご参照ください。

ご注意：(1)取扱説明書に記載された内容以外でご使用された場合、機能・性能の保証はできませんのでご注意ください。
(2) 本書に掲載されている会社名や製品は、一般に各社の商標または登録商標です。

本書が扱う対象製品

パナソニック デバイスＳＵＮＸ(株)製　PV200／PV500 シリーズ

お願い

ご使用前に「マニュアル・安全上のご注意」をお読みいただき正しく安全にプロバイダをご使用下さい。

お客さまへ

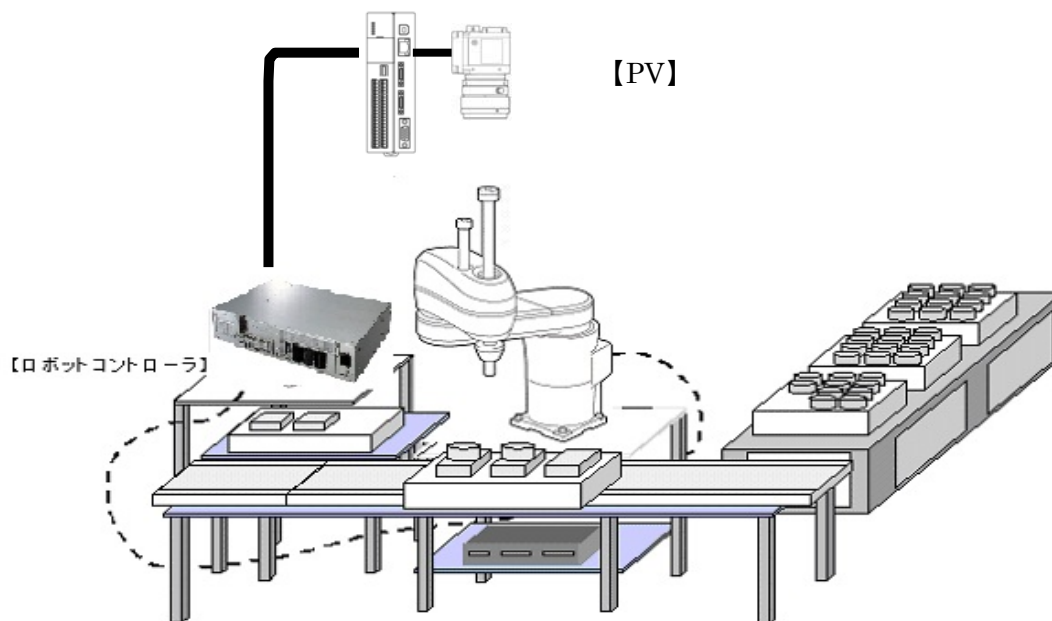
1.　ご使用に係わるリスクについて

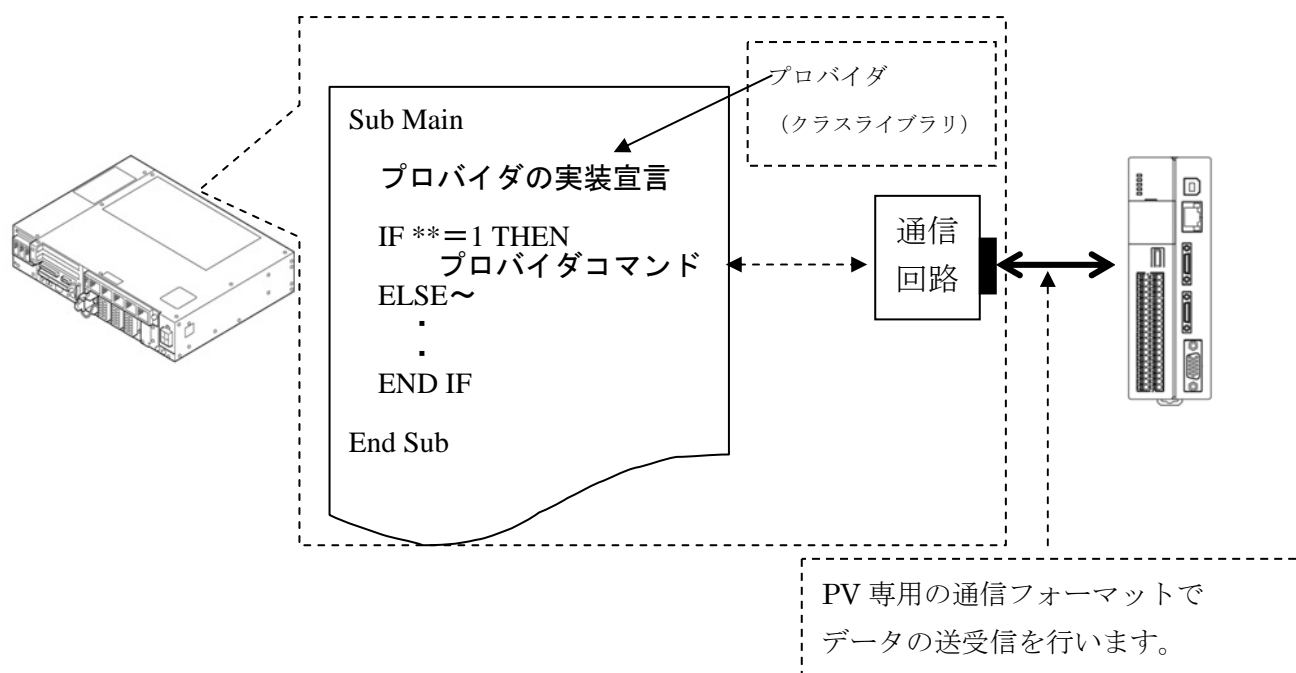
本製品（ソフトウェア）のシステム組み込み・使用ならびに本製品の使用結果に関する一切のリスクは、本製品の使用者に帰属するものとします。

目次

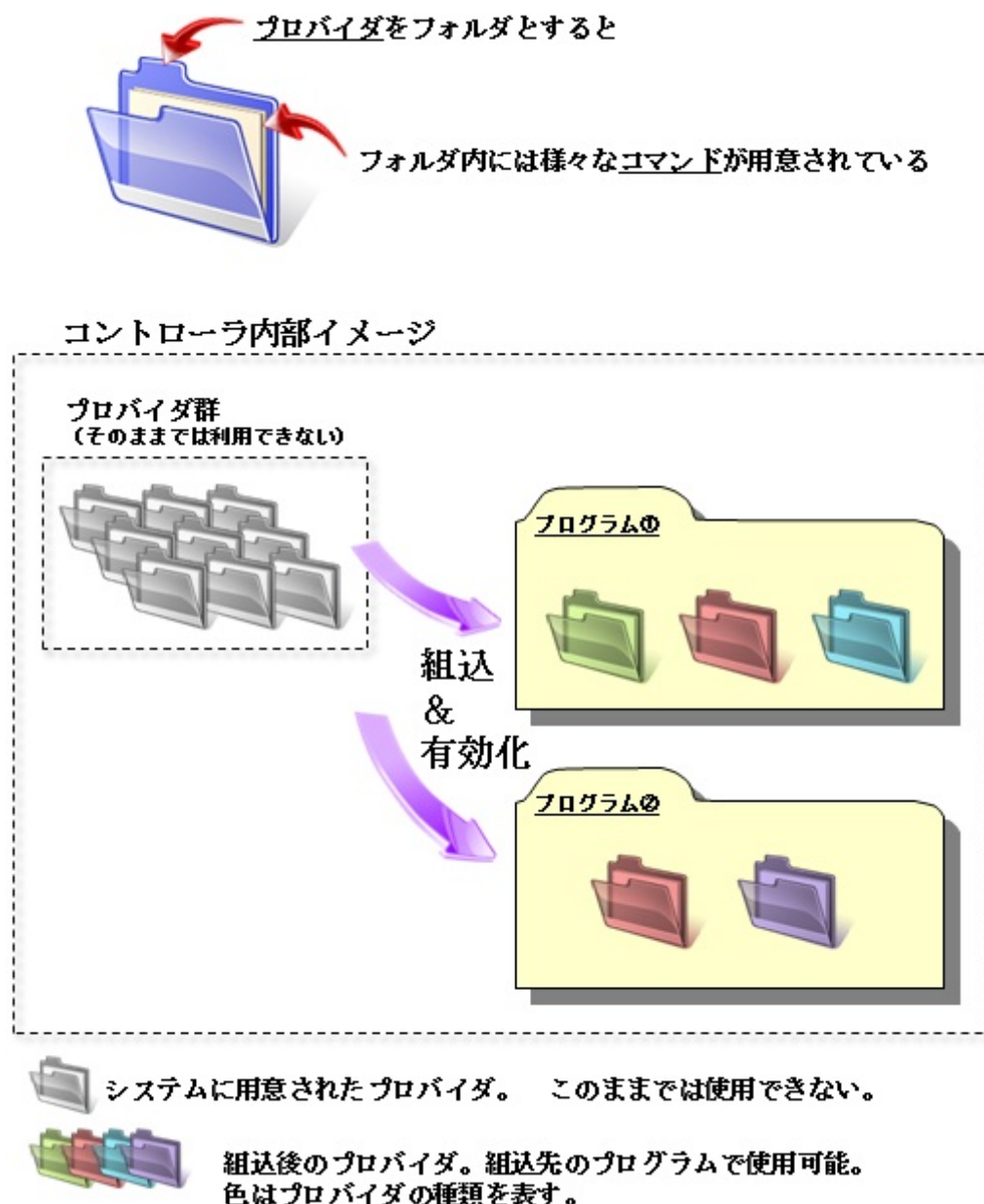
1.　本製品（プロバイダ）の概要

1.1 プロバイダの対象機器

本プロバイダは、デンソーロボットコントローラ（RC8シリーズ）とＰＶシリーズを接続する時に使用することができます。

1.2 プロバイダの特長

ＰＶシリーズにアクセスするために必要なＰＶのネイティブコマンドを、ロボットプログラムで使用できるプロバイダとして準備しています。本プロバイダを使用することで、ＰＶシリーズの通信部分のプログラムを組むことなく、容易にロボットからの通信を行うことができます。下記にプロバイダの組込関係を示します。

1.3 プロバイダの仕組み

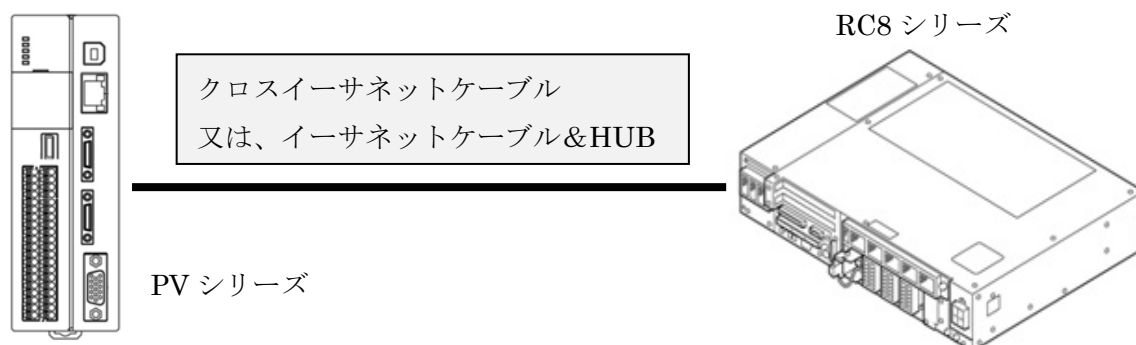
本プロバイダは対象製品の制御を行うための各種プログラムをひとつのプロバイダとして提供しています。使用にあたってはライセンスを有効化するだけで使用する事が出来ます。使用したいプログラムファイルで実装の宣言をすれば、それ以降はプロバイダが用意する関数をコマンドとしてユーザープログラム内で使用する事が可能です。本プロバイダはコントローラ内に予め用意されていますので、インストール作業は不要です。又、違う種類のプロバイダであれば複数個実装する事も可能です。但し、同じ種類のプロバイダは同じプログラム（プロシージャ）内で存在する事は出来ません。

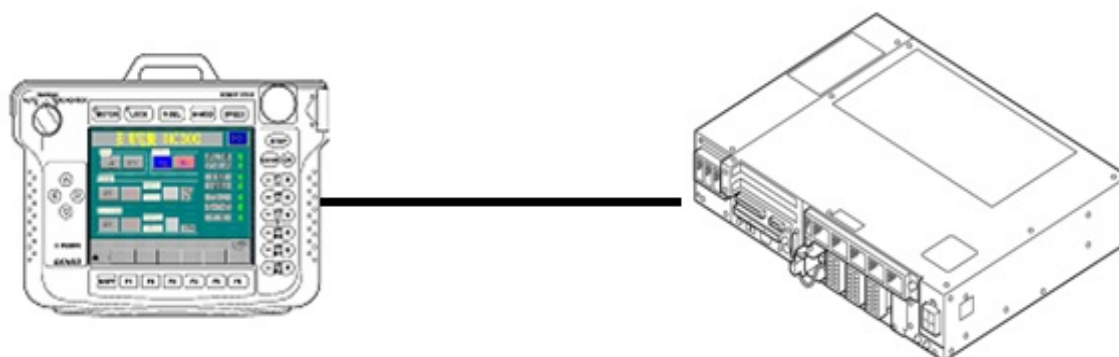
組込後のプロバイダ。組込先のプログラムで使用可能。
色はプロバイダの種類を表す。

注：上図のプロバイダ　の様に、同一のプロバイダがプログラム別に存在する場合は、プログラム（タスク）間で排他処理を行う必要があります。

※プロバイダは PacScript から使用できるダイナミックリンクライブラリ（以下 DLL）として用意されています。

2. 接続方法

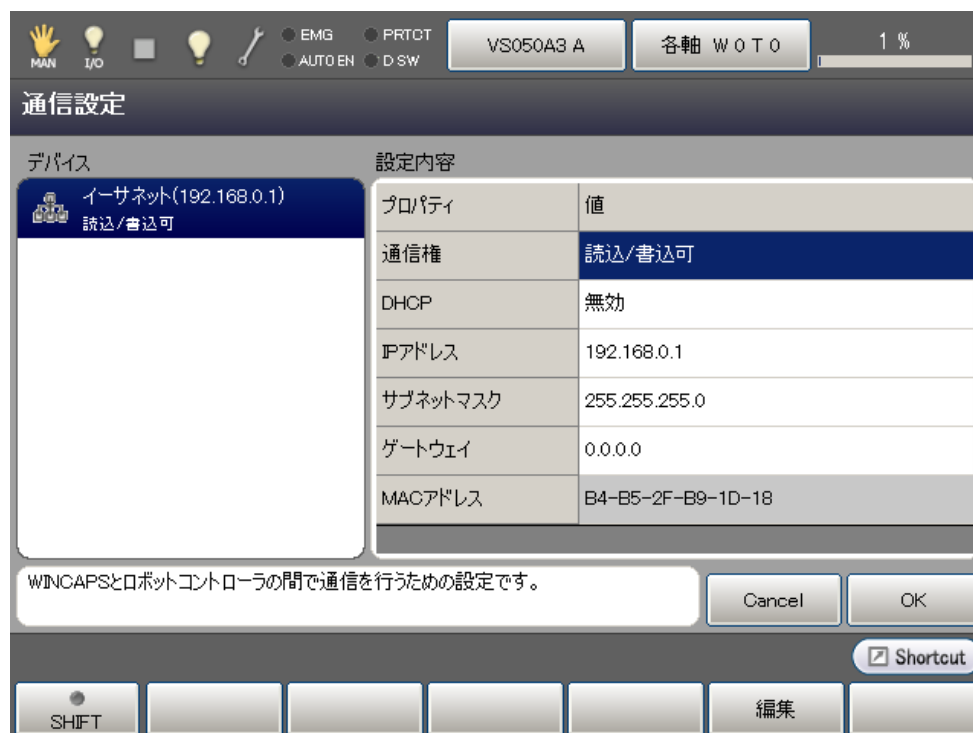
2.1 Ethernet（TCP/IP）接続例

ロボットコントローラとPVシリーズをEthernet(TCP/IP)接続する際には、クロスイーサネットケーブルをご使用下さい。又、スイッチングハブ/ルータを使用する場合は、スイッチングハブ/ルータの仕様に合ったケーブルをご使用下さい。

3. ロボットコントローラと使用機器の通信設定

ティーチングペンダントを使用して、各通信設定項目を使用機器に合わせて下さい。

3.1 Ethernet（TCP/IP）による通信

3.1.1 ロボットコントローラのEthernet（TCP/IP）通信設定

ロボットコントローラのIPアドレスを設定します。

［F6設定］［F5通信と起動権］［F2ネットワークと通信権］で、通信設定ウィンドウが表示されます。ロボットコントローラとPVが、同一のサブネットマスク内になるようにIPアドレス及び、サブネットマスクを設定して下さい。

3. 2. 2 PVのEthernet（TCP/IP）通信設定

PVのIPアドレスを設定します。

[ツール]－[ネットワーク設定]を選択します。ネットワーク設定画面で IP アドレスとサブネットマスクを設定して下さい。ロボットコントローラと PV が、同一のサブネットマスク内になるように IP アドレス及び、サブネットマスクを設定して下さい。

【PV の画面】

3. 2. 3 PVの結果出力設定

[環境]－[入出力]－[汎用結果出力]を選択します。Ethernetの汎用出力（プロトコル）の設定を行って下さい。なお、詳細な設定については、パナソニック社製PVユーザーズマニュアルを参照下さい。

	Serial	Ethernet	Ethernet	SD Card
出力	しない	しない	する	しない
出力動作	同期	同期	同期	同期
プロトコル	汎用通信	PLC通信	汎用通信	
日付・時刻			しない	しない
走査回数	しない	しない	しない	しない
総合判定	しない	しない	しない	しない
判定出力	しない	しない	しない	しない
数値演算	する	する	する	する
BCC	しない		しない	
出力桁数	5		5	5
小数点以下桁数	2		2	2
無効桁出力	削除		削除	削除
エラー出力			しない	しない

4. プロバイダ実行手順

プロバイダは実装(宣言)→実行が基本の手順になります。本プロバイダは実装時に接続処理を行います。操作は必要な分だけ繰り返す事が出来ます。プログラム例を下記に示します。

```
Sub Main

    On Error Goto ErrorProc        ①        '異常処理ルーチンの宣言
    Dim caoCtrl as Object          ②        'プロバイダ用変数宣言
    Dim strResult as String        ③        '結果取得用変数宣言

    caoCtrl = cao.AddController("PV", "caoProv.Panasonic.PV", "", "conn=eth:192.168.0.201") ④

    「トリガ　～　データ受信処理を記述」⑤

EndProc:
    '終了処理
    Exit Sub

ErrorProc:
    '異常処理

End Sub
```

① 必要があればプロバイダ異常時の処理ルーチンを宣言します。(宣言時の接続異常検出)

② プロバイダを実装させる変数を Object 型で宣言します。変数名は任意に指定できます。

③ 結果を取得する変数を宣言します。データ型はコマンドにより違います。

④ プロバイダ宣言コマンド cao.AddController で実装します。設定に必要なパラメータはプロバイダで違います。これ以降は実装変数 caoCtrl を利用してプロバイダコマンドを利用できます。

⑤ これ以降は、プロバイダコマンドを使用したプログラムが記述可能です。

5. コマンドの説明

本章では各コマンドについて説明します。コマンドは接続コマンド、PV コマンド、独自拡張コマンドに分類されます。尚、PV コマンドの詳細動作については Panasonic 社の PV シリーズのマニュアル汎用通信コマンド詳細を参照してください.

表 5-1 コマンド一覧

コマンド	PV コマンド	機能	参照
接続コマンド			
cao.AddController	―	プロバイダを変数に実装して、PV に接続処理を行います	11
Addvariable	―	画像やセルの値を取得する為の専用変数を作成します	12
Value	―	AddValiable で作成した変数を経由して取得します	13
PV コマンド			
Start	%S	検査実行（一括トリガ）	14
Start	%S	繰り返し実行の停止	14
Start	%S	検出トリガ実行の停止	14
Restart	%R	再検査実行（撮像せずに現在のメモリ画像で検査する）	15
Xtype	%X	品種切り替え	16
MemoryWrite	%MW	設定データ保存　本体保存用メモリ	17
CFWrite	%CW	設定データ保存　SD カードメモリ	18
MemoryRead	%MR	設定データ読み出し　本体保存用メモリ	19
CFRead	%CR	設定データ読み出し　SD カードメモリ	20
CancelData	%CD	設定データ保存/読み出しの中断（キャンセル）	21
SDSave	%SS	保存画像メモリ　保存(SD メモリーカード)	22
SDReset	%SR	保存画像メモリ　消去	23
PrintScreen	%PS	プリントスクリーン	24
Quit	%Q	統計リセット	25
RunManual	%RM	運転/停止（RUN/STOP）の切り替え	26
ErrorReset	%E	エラー信号のリセット	27
Cancel	%CC	検査/処理の中断（各種動作キャンセル）	28
KeyEmulator	%K	キーエミュレート	29
Bstop	%BS	キーパッド操作　受付禁止/受付許可	30
Bconfirm	%BC	キーパッド操作　受付状態の確認	31
LayOutChange	%I	レイアウト切り替え	32
AgainTemplate	%A	テンプレートの再登録	33
ParameterRead	%PR	パラメータ　読み出し	34
ParameterReadPair	%PRP	パラメータ　ペア読み出し（各種上下限値など）	35
ParameterWrite	%PW	パラメータ　変更	36
ParameterWritePair	%PWP	パラメータ　ペア変更（各種上下限値など）	37
独自コマンド			
Raw	―	コマンドメッセージの送受信	38

cao.AddController

機能

プロバイダを変数に実装して、PV に接続処理を行います。

書式

cao.AddController(〈コントローラ名〉,〈プロバイダ名〉,
〈プロバイダの実行マシン名〉,〈オプション〉)

引数：
〈コントローラ名〉任意名を付けて下さい（名前で管理しています）
〈プロバイダ名〉 “CaoProv.Panasonic.PV”
〈プロバイダの実行マシン名〉 省略して下さい
〈オプション〉［接続パラメータ］,［タイムアウト時間］

［接続パラメータ］ “conn=eth:〈IP アドレス〉”
［タイムアウト時間］ 送受信時のタイムアウト時間(msec)を指定します。
“Timeout[=時間]”。デフォルト：500。

説明

プロバイダを変数に実装すると同時に有効にします。これ以降は実装した Object 型変数を使用してプロバイダにアクセスします。（実装された変数を“実装変数”と呼びます。）

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("PV", "caoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
```

タイムアウトを指定したい場合は次のように記述します

```
caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201, Timeout = 1000")
```

実装変数.AddVariable

機能

画像を取得する為のプロパティ変数を作成します。

書式

実装変数.AddVariable(<画像指定>, [<オプション>])

引数： <画像指定> 取得画像の種類を指定します。
@BITMAP：カメラ画像
@BITMAP_MONITOR：モニタ画像
<オプション> 無し

説明

PV から画像を取得する為の変数を Object 型で作成します。

用例

画像を取得し，操作盤の画面に表示する例を示します。

```
Dim caoCtrl as Object
Dim objBmp as Object

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")

objBmp = caoCtrl.AddVariable("@BITMAP", "")
Label.Picture = objBmp.Value
```

プロパティ変数.Value

機能

画像データを、AddValiable で作成した変数を経由して取得します。

書式

プロバイダ変数.Value

戻り値：　BITMAP フォーマットデータ。

説明

プロバイダ（実装変数）から画像データを、AddVariable で作成した変数を経由して取得します。

用例

画像を取得し、操作盤に表示する例を示します。

```
Dim caoCtrl as Object
Dim objBmp as Object

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")

objBmp = caoCtrl.AddVariable("@BITMAP", "")
Label.Picture = objBmp.Value
```

実装変数.Start

機能

検査を実行します。実行モードが「全実行」または「分岐実行」の場合と、「指定実行」の場合で書式が異なります。画像処理結果は PV シリーズの「汎用結果出力」で設定した値を文字列で返します。

書式

実装変数.Start(［ブロックナンバー］ **)**

引数：［ブロックナンバー］　実行するブロックナンバー（整数 0～9）

戻り値：画像処理結果（文字列）

説明

一括トリガで実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］、「分岐実行」の場合、引数は不要です。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
strResult = caoCtrl.Start( 2 )          ‘指定実行

strResult = caoCtrl.Start               ‘全実行または分岐実行
```

実装変数.Restart

機能

画像取り込みをせずに検査を実行します。実行モードが［全実行］または［分岐実行］の場合と、「指定実行」の場合では書式が異なります。

書式

実装変数.Restart(［ブロックナンバー］**)**

引数：［ブロックナンバー］　実行するブロックナンバー（整数 0～9）

戻り値：画像処理結果（文字列）

説明

一括トリガで実行モードが［指定実行］の場合のみ引数のブロックナンバーが必要です。
［全実行］、「分岐実行」の場合、引数は不要です。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
strResult = caoCtrl.Restart( 2 )        '指定実行

strResult = caoCtrl.Restart             '全実行または分岐実行
```

実装変数.Xtype

機能

品種を切り替えます。

書式

実装変数.Xtype 〈品種ナンバー〉

引数：〈品種番ナンバー〉 （整数 0～255）

説明

品種を切り替えます。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.Xtype 11
```

実装変数.MemoryWrite

機能 設定データを本体メモリに保存します。

書式 **実装変数.MemoryWrite**

引数:<エリアナンバー> SD メモリカードの保存エリアナンバーを指定します。(整数 0～99)

説明 設定データを本体メモリにエリアナンバーを指定して保存します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.MemoryWrite
```

実装変数.CFWrite

機能 設定データを SD メモリカードへ保存します。

書式 **実装変数.CFWrite** ＜エリアナンバー＞

引数:〈エリアナンバー〉SD メモリカードの保存エリアナンバーを指定します。
(整数 0～99)

説明 設定データを SD メモリカードへエリアナンバーを指定して保存します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.CFWrite 4
```

実装変数.MemoryRead

機能

本体メモリから設定データを読み出します。

書式

実装変数.MemoryRead ＜エリアナンバー＞

引数：〈エリアナンバー〉SD メモリカードの読み出しエリアナンバーを指定します。（整数 0～99）

説明

本体メモリからエリアナンバーを指定して設定データを読み出します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController( “PV” , “CaoProv.Panasonic.PV” , “” , “conn=eth:192.168.0.201” )
caoCtrl.MemoryRead
```

実装変数.CFRead

機能

SD メモリカードから設定データを読み出します。

書式

実装変数.CFRead ＜エリアナンバー＞

引数：〈エリアナンバー〉SD メモリカードの読み出しエリアナンバーを指定します。（整数 0～99）

説明

SD メモリカードからエリアナンバーを指定して読み出します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController( “PV” , “CaoProv.Panasonic.PV” , “” , “conn=eth:192.168.0.201” )
caoCtrl.CFRead 4
```

実装変数.CancelData

機能 設定データの保存・読み出しを中断します。

書式 **実装変数.CancelData**

説明 設定データの保存・読み出しを中断します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController( “PV”， “CaoProv.Panasonic.PV”， “”， “conn=eth:192.168.0.201” )
caoCtrl.CancelData
```

実装変数.SDSave

機能 本体に保存されている画像メモリを SD メモリカードに保存します。

書式 **実装変数.SDSave**

説明 本体に保存されている画像メモリを SD メモリカードに保存します。
ＳＤメモリカードの保存先は，空き番号を探して保存します。（保存先の番号は指定できません）

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController( “PV” , “CaoProv.Panasonic.PV” , “” , “conn=eth:192.168.0.201” )
caoCtrl.SDSave
```

実装変数.SDReset

機能 本体に保存されている画像メモリを消去します。

書式 **実装変数.SDReset**

説明 本体に保存されている画像メモリを消去します。
　設定画面で［保存・読出］→［画像メモリ消去］を実行したときと同じ動作です。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.SDReset
```

実装変数.PrintScreen

機能

現在の画面表示（表示されているものすべて）をキャプチャし、SD メモリカード、または Ethernet インターフェイス経由でパソコンに保存します。

書式

実装変数.PrintScreen

説明

現在の画面表示をキャプチャし、SD メモリカード、または Ethernet インターフェイス経由でパソコンへ保存します。
保存先は［環境］→［入出力］→［プリントスクリーン］の［出力先］で指定した場所です。このコマンドでは出力先を指定できません。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.PrintScreen
```

実装変数.Quit

機能 統計データと走査回数をクリアします。

書式 **実装変数.Quit**

説明 統計データと走査回数をクリアします。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController( “PV” , “CaoProv.Panasonic.PV” , “” , “conn=eth:192.168.0.201” )
caoCtrl.Quit
```

実装変数.RunManual

機能

PV シリーズの運転・停止を切り替えます。

書式

実装変数.RunManual(＜モード＞)

引数：〈モード〉　運転・停止の切り替え（整数）。
0：運転へ切り替え。
1：停止へ切り替え。
戻り値：切り替えたモードの値（整数）。
2：運転。
12：停止。

説明

PV シリーズの運転・停止を切り替えます。

用例

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController(“”, “CaoProv.Panasonic.PV”, ,””, “conn=eth:192.168.0.201”)
iResult = caoCtrl.RunManual( 0 )
```

実装変数.ErrorReset

機能 Error 信号をリセットします。

書式 **実装変数.ErrorReser**

説明 Error 信号をリセットします。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.ErrorReset
```

実装変数.Cancel

機能 実行途中の動作をキャンセルします。

書式 **実装変数.Cancel**

説明 実行途中の動作をキャンセルし、その動作の開始前の状態にします。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.Cancel
```

実装変数.KeyEmulator

機能

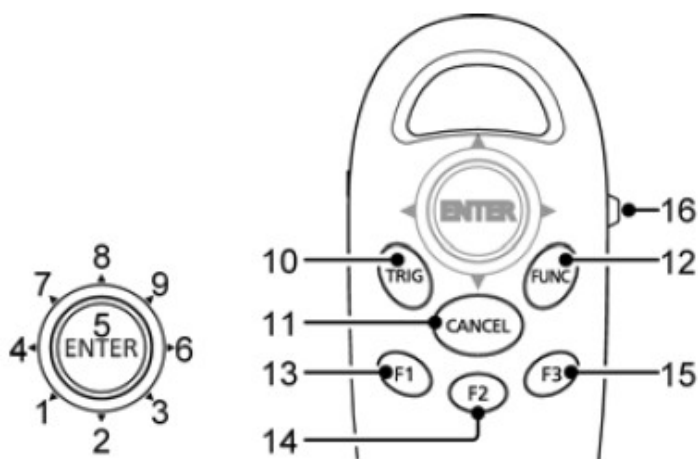
キーパッドと同じ操作を実行します。

書式

実装変数.KeyEmulator ＜シフト＞，＜キー＞

引数：〈シフト〉　シフトキーの ON・OFF（整数 0, 1）。
0：OFF。
1：ON。
〈キー〉　各種キーに割り当てられた値（整数 1～16）。
詳細は下図参照。

説明

キーパッドと同じ操作を実行します。
PV シリーズからのレスポンスはありません。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.KeyEmulator 0, 11
```

実装変数.Bstop

機能

運転画面において、キーパッドによる操作の受付を禁止・許可します。

書式

実装変数.Bstop　＜可否＞

引数：<可否>　キーパッド操作の受付けの可否（整数 0, 1）。
0：受付許可。
1：受付禁止。

説明

運転画面において、キーパッドによる操作の受付を禁止・許可します。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“”, “CaoProv.Panasonic.PV”, ,””, “conn=eth:192.168.0.201”)
caoCtrl.Bstop 1
```

実装変数.Bconfirm

機能　　キーパッドによる操作の受付状態を取得します。

書式　　**実装変数.Bconfirm**

戻り値：　キーパッド操作受付状態（整数 0, 1)。
0：受付許可。
1：受付禁止。

説明　　キーパッドによる操作の受付状態を取得します。

用例

```
Dim caoCtrl as Object
Dim iResult as Integer

caoCtrl = cao.AddController(“”, “CaoProv.Panasonic.PV”, ,””, “conn=eth:192.168.0.201”)
iResult = caoCtrl.Bconfirm
```

実装変数.LayoutChange

機能

運転画面でモニタ表示するレイアウトを切り替えます。

書式

実装変数.LayoutChange ＜レイアウトナンバー＞

引数：〈レイアウトナンバー〉　整数で指定（0～15）。

説明

運転画面でモニタ表示するレイアウトを切り替えます。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.LayoutChange 7
```

実装変数.AgainTemplate

機能

スマートマッチングチェカーのテンプレートを再登録します。

書式

実装変数.AgainTemplate　＜チェッカーナンバー＞
，＜テンプレートナンバー＞

引数：〈チェッカーナンバー〉　整数で指定（0～999)。
〈テンプレートナンバー〉　整数で指定（0～63)。

説明

再登録できるのは［チェッカ］下のスマートマッチングです。位置補正、領域調整で使っているスマートマッチングは、テンプレートの再登録は出来ません。

用例

```
Dim caoCtrl as Object

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.AgainTemplate 1, 10
```

実装変数.ParameterRead

機能

PV シリーズ本体の設定値やシステム値を読み出します。

書式

実装変数.ParameterRead(　＜パラメータ＞)

引数 : 〈パラメータ〉　文字列で指定。

戻り値 : 指定したパラメータ値。（文字列）

説明

PV シリーズ本体の設定値やシステム値を読み出します。運転中にのみ有効です。読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

```
Dim caoCtrl as Object
Dim strResult as String

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
strResult = caoCtrl.ParameterRead( “SYS_TIME1” )
```

実装変数.ParameterReadPair

機能 PV シリーズ本体の設定値やシステム値の 2 データを読み出します。

書式 **実装変数.ParameterReadPair(** ＜パラメータ＞ **)**

引数：〈パラメータ〉 文字列で指定。

戻り値：指定したパラメータ値。(Variant 型)

説明 PV シリーズ本体の設定値やシステム値を 2 データ読み出します。上下限値データなどのセットデータを読み出します。運転中にのみ有効です。読み出し可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

```
Dim caoCtrl as Object
Dim vntResult as Variant

caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
vntResult = caoCtrl.ParameterReadPair( "BLV:PAIRA" )
```

実装変数.ParameterWrite

機能　　PV シリーズ本体の設定値やシステム値を変更します。

書式　　**実装変数.ParameterWrite**　＜パラメータ＞，＜データ＞

引数：〈パラメータ〉　文字列で指定。
　　　〈データ〉　Variant 型で指定。

説明　　PV シリーズ本体の設定値やシステム値を変更します。運転中にのみ有効です。変更可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

```
Dim caoCtrl as Object
Dim vntData as Variant

vntData = 3.14
caoCtrl = cao.AddController("PV", "CaoProv.Panasonic.PV", "", "conn=eth:192.168.0.201")
caoCtrl.ParameterWrite "SYS:REG0", vntData
```

実装変数.ParameterWritePair

機能

PV シリーズ本体の設定値やシステム値を 2 データ変更します。

書式

実装変数.ParameterWritePair ＜パラメータ＞
　　, ＜データ 1＞
　　, ＜データ 2＞

引数 : 〈パラメータ〉　文字列で指定。
〈データ 1〉　Variant 型で指定。
〈データ 2〉　Variant 型で指定。

説明

PV シリーズ本体の設定値やシステム値を 2 データ変更します。運転中にのみ有効です。変更可能なデータ、各種コマンドパラメータについては、Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

```
Dim caoCtrl as Object
Dim vntData1 as Variant
Dim vntData2 as Variant

vntData1 = 50
vntData2 = 100
caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
caoCtrl.ParameterWritePair “BLV:PAIRA”, vntData1, vntData2
```

実装変数.Raw

機能

コマンドメッセージを送受信します。

書式

実装変数.Raw(＜送信コマンドメッセージ＞ **)**

引数：〈送信コマンドメッセージ〉　文字列で指定。

戻り値：受信したコマンドメッセージ。（文字列）

説明

PV シリーズのコマンドを直接送受信します。BCC（ブロックチェックコード）に付いては内部で自動計算します。
コマンドについては Panasonic 社の PV シリーズマニュアルを参照して下さい。

用例

```
Dim caoCtrl as Object
Dim strResult as strResult

caoCtrl = cao.AddController(“PV”, “CaoProv.Panasonic.PV”, “”, “conn=eth:192.168.0.201”)
strResult = caoCtrl.Raw(“%S”)
```

6. 操作盤画面

本プロバイダには下記の操作盤画面を準備しています。この操作盤はプロバイダを使用したもので、機器接続後の動作確認等に使用することができます。操作盤のアプリケーション例の参考にして下さい。操作盤を表示するとPVへ接続（プロバイダの実装）をしますので予め通信設定を行って下さい。操作盤を閉じると切断（プロバイダの解放）されます。

【メイン面】

説明 各ボタンの動作内容について説明します。

1. 「全実行」または「分岐実行」へ切り替えます。
2. 「指定実行」へ切り替えます。
3. 変更する品種を設定します。範囲：0～255
4. 3. で設定した品種に切り替えます。（Xtype）
5. 「指定実行」時のブロック No.を設定します。範囲：0～9
6. 3. 5. で設定した番号に従い検査を実行します。受信したデータは 9. のデータ表示部に表示します。(Start)
7. 処理結果の状態を表示します。
8. 受信データの表示ページを Up します。
9. 受信データを表示します。
10. 受信データの表示ページを Down します。

注１：プロバイダの実装（初期化）が正常に行われた場合は、7. に“接続完了”と表示されます。

注２：PacScript プログラムにて PV プロバイダを使用している場合は、操作盤操作をしないで下さい。

7. サンプルプログラム

```
Sub Main

    On Error Goto ErrProc                      ‘異常処理ルーチン宣言

    Dim caoPV as Object                        ‘プロバイダ用変数宣言
    Dim strResult as String                    ‘文字列変数宣言
    Dim pTargetPos as Position                 ‘P 変数型変数宣言

    takearm keep = 0

    pTargetPos = P11

    caoPV = cao.AddController(“PV”, “CaoProvider.Panasonic.PV”, “”, “Conn=eth:192.168.0.110, Timeout
= 1000”)                                       ‘プロバイダの実装
    caoPV.Xtype 2                              ‘品種 2 に切換
    strResult = caoPV.Start                    ‘トリガ→処理待ち
    letx pTargetPos = posx(P11) + val(strResult)   ‘受信データの X 成分を位置データへ展開

    approach p, pTargetPos, @p 20, s = 100     ‘補正後の位置へ
    move l, @e pTargetPos, s = 10
    call Hand.Close
    depart l, @p 50, s = 100

EndProc:                                       ‘正常終了ルーチン
    「必要な終了処理を記述」
    exit sub

ErrProc:                                       ‘異常終了ルーチン
    「必要な異常処理を記述」

End Sub
```

改訂履歴

デンソーロボット
プロバイダ
取扱説明書
パナソニックデバイス SUNX(株)製　ビジョンセンサ　PV シリーズ

バージョン	対応RC8	改訂内容
Ver.1.0.0	Ver.1.1.2	初版
Ver.1.0.1	Ver.1.3.6～	変数追加　“@ResultDisable”
Ver.1.0.2	Ver.1.3.7～	RunManualコマンド修正

株式会社デンソーウェーブ